User Request Provisioning Oriented Slice Anomaly Prediction and Resource Allocation in 6G Networks

Zhao Ming*, Hao Yu*, and Tarik Taleb†
*Oulu University, Oulu, Finland.
†Ruhr University Bochum, Bochum, Germany
Email: {zhao.ming, hao.yu}@oulu.fi, tarik.taleb@rub.de

***Abstract*—Satisfying users' requests based on the service level agreements of network slices is one of the most basic and vital topics of network slicing in 6G networks, and anomaly detection is regarded as a key technique for locating the abnormal status of slices. However, current studies on slice anomaly detection mostly focused on real-time monitoring of slices and ignored the prediction of potential anomalies. Generally, when anomalies trigger, it is hard for slices to adjust the resources in time due to resource competition among physical/virtual nodes. Besides, the resource provisioning strategies can also be optimized when slices are running normally, which is seldom considered when performing slice anomaly detection. To cope with these challenges, in this paper, we are motivated to locate the potential slice anomalies and optimize the resource allocation strategies in a holistic view by learning users' historical behaviors. Specifically, we design a general network architecture, model the process of slice resource provisioning, and formulate the problem as maximizing the long-term system net promoter score (NPS). To solve this problem, we propose a framework to locate the potential slice anomalies and decide the resource allocation strategies simultaneously by predicting the users' future requests and positions. As a result, simulation results demonstrate that our proposed scheme outperforms other baselines in improving the long-term system NPS and reducing the average latency of users.**

I. Introduction

Network slicing, enabled by the network functions virtualization and software-defined networking technologies, is regarded as a key technology for supporting heterogeneous user requests in the 5G era by separating the networks into specific logical sub-networks and isolating the users in different scenarios including the Internet of things/vehicles, smart healthcare, and smart city [1], [2]. In future 6G systems, much stricter network requirements like increased data rates, enhanced network capacities, ultra-low latency preferences, and more connected devices make it necessary to dig deeper in this field [3].

Satisfying users' requests to ensure service level agreements (SLAs) of network slices is one of the most important topics among the research visions of network slicing. However, constructing on top of physical networks and serving multiple users also brings significant challenges for the management and orchestration of network slices. On the one hand, the topology and resource management of network slices can be complex, as they not only depend on a multilayered physical network and edge servers (ESs) with limited resources, but can also be dynamically adjusted (e.g., slice splitting, merging, and scaling) [4]–[8]. On the other hand, slice users are always characterized by high mobility and dynamical requests, satisfying users' requests with the committed SLAs in a real-time manner requires the slices to dynamically perceive users' requests and adjust the resource provisioning strategies. To cope with these challenges, performing anomaly detection in network slices is regarded as a key solution [9], [10]. By monitoring the running status of physical/virtual nodes, the connectivities of physical/virtual links, and the latencies of different service function chains in the sub-slices level and slice level, slice anomaly detection can improve the users' quality of service/experience (QoS/QoE) by ensuring users' specific requests for resources, service latencies, computing capacities, and content availabilities.

Many researchers have studied network slice anomaly detection in recent years, which can be elaborated as two-fold. In [10], Wang *et al.* proposed a distributed online anomaly detection method based on a decentralized one-class support vector machine by analyzing real-time data of virtual nodes mapped to physical nodes. The correlations of data between neighbor virtual nodes were adopted to detect the anomalies in physical links. In [11], the authors proposed an online unsupervised learning-driven slicing anomaly detection method to achieve intelligent maintenance of slices. However, these studies ignored the prediction of potential anomalies in network slices, when anomalies trigger, it is generally hard to adjust the resources in time due to resource competition among physical/virtual nodes, which will unavoidably lead to service degradation of users. On the other hand, in [12], the authors designed a cognitive network slice management system that utilizes anomaly detection to detect the anomalies that occur in the ambulance routes with the required QoS level considered. Moreover, prediction-based methods were also utilized to assist anomaly detection. The researchers in [9] proposed a transfer learning-based framework to detect the anomalies in shared physical nodes in network slices. Even though these studies can detect/predict the anomalies that affect the SLAs of slices, they also ignore the optimization of resource provisioning strategies when locating the anomalies. Note that when the slices run normally with users' requested resources satisfied, the users' QoE is associated with the latencies of provided resources and can be further optimized for better QoE performances.

To cope with these issues, in this paper, we are motivated to investigate the anomaly prediction and resource allocation of network slices by learning users' historical behaviors to

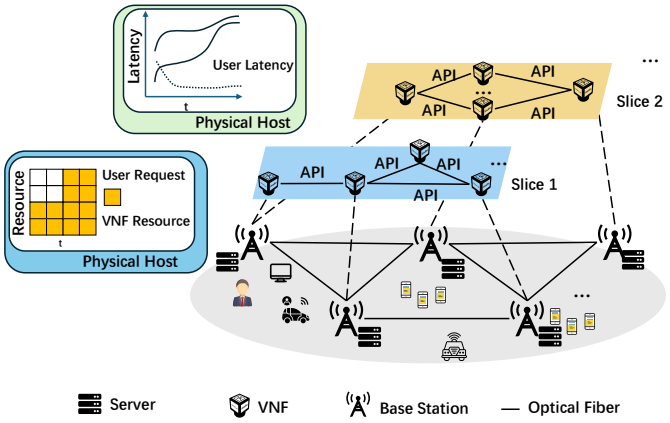


Fig. 1. The considered resource provisioning network architecture in 6G systems.

locate the potential threats of network slices and improve users' QoE. Specifically, we design a general network architecture, model the resource provisioning process with users' requested resources and preferred latencies considered, and formulate the problem as maximizing the long-term system net promoter score (NPS). To solve this problem, we propose a resource provisioning-oriented slice anomaly prediction and resource allocation framework that learns users' historical behaviors to locate the potential anomalies and allocate resources for users according to their requests. Finally, based on the simulation results, we demonstrate that the proposed framework can significantly improve the long-term system NPS and reduce the average latency of users compared to other baseline schemes. To summarize, the contributions of this paper are as follows:

- We investigate the anomaly prediction and resource allocation of slice resource provisioning, design a general network architecture, and formulate the problem as maximizing the long-term system NPS to improve users' QoE.
- To solve this problem, we propose an anomaly prediction and resource allocation framework that learns users' historical behaviors to detect potential anomalies and decide the resource allocation strategies.
- Simulation results demonstrate that our proposed scheme outperforms other baselines in improving the long-term system NPS and reducing the average latency of users.

The remainder of this paper is organized as follows. Sect. II introduces the network architecture and system model and formulates the problem. In Sect. III, we propose a deep learning-based anomaly prediction and resource allocation framework. Simulation results are provided in Sect. IV. Finally, Sect. V concludes this paper.

II. System Model and Problem Formulation

In this section, we first introduce the considered network architecture for user request provisioning, model the whole process, and finally formulate the problem.

A. Network Architecture

We illustrate a general network topology for supporting heterogeneous user requests in 6G networks as shown in Fig. 1. Specifically, in the considered networks, there exist lots of base stations (BSs) geographically evenly distributed and connect to their neighbor BSs through optical links, and multiple user equipments (UEs) like smartphones and vehicles connect to the BSs via wireless links. Each BS is equipped with a small ES that has limited resources, e.g., CPU cores, RAM, or disk resources, and the UEs with high mobility characteristics can dynamically move to different places and timely change their requested resources with specific latency preferences. We assume that the users have specific requirements of latency for different types of resources, e.g., users may prefer a large amount of disk resources with high latency for storage or RAM resources with low latency for running multiple services.

To support these requests from users, different network slices are then initialized based on the characteristics of the requests, e.g., the slices that serve users who request lots of resources with high latency preferences tend to allocate more resources and low bandwidth and vice versa. These slices are generally built with virtual network functions (VNFs) that are connected partially via virtual links, where each VNF has limited resources and can only be hosted by one ES. After the initialization, the users dynamically move and change their requested resources and latencies, and the slices decide the resource provisioning strategies to satisfy the users' requests. Once the users' requests cannot be satisfied, the slices then occur anomalies and adjust the topologies, the allocated resources, and even the hosts of the VNFs to form new slices for resource provisioning. Note that the adjustment of slices can lead to users' QoE degradation and to ensure the resource provisioning of slices, in this paper, we try to predict the anomalies of network slices and decide the resource allocation strategies when no anomaly occurs to improve the QoE of users.

B. System Model

We denote the ESs and users in the system as $\mathcal{N}$ and $\mathcal{U}$, respectively, the geographic coordinator of ES $n \in \mathcal{N}$ can be denoted as X_n, Y_n. Moreover, as the users dynamically move to different places, we denote the position of user u at time slot t as $X_u(t), Y_u(t)$. We consider there are I types of requested resources from the users like CPU, RAM, disk *et al.*, and denote the total amount of resource type i of ES n as R_n^i. Particularly, considering each user has a specific required latency of each resource type, we denote the requested amount of resource type i from user u at time slot t as $R_u^i(t)$, the requested latency as $L_u^i(t)$, respectively. To support heterogeneous user-requested resources, the ESs initialize network slices for different groups of users according to their required resources/latency levels[1], the set of slices is denoted as $\mathcal{S}$. Moreover, we denote the set of users served by s as $\mathcal{U}_s$, the set of VNFs of s as $\mathcal{V}_s$, and the total amount of resource type i of VNF $v \in \mathcal{V}_s$ as R_v^i.

[1]In this paper, we focus on anomaly prediction and resource allocation of network slices and neglect the details for slice initialization.

When user $u \in \mathcal{U}_s$ dynamically moves across different places, to access the resources from the slice s that serves it, the slice can allocate resources from one or multiple VNFs for the user. We denote the allocated amount of resource type i from $v \in \mathcal{V}_s$ for user u at time slot t as $R^i_{u,v}(t)$, and the access latency is denoted as $l^i_{u,v}(t)$. Intuitively, if the access latency $l^i_{u,v}(t)$ cannot satisfy the request latency from the user, we can have $R^i_{u,v}(t) = 0$. To calculate $l^i_{u,v}(t)$, for any $v \in \mathcal{V}_s$, we denote the ES host of v as $h_v \in \mathcal{N}$. To access resource i from VNF v, user u should first connect to the nearest VNF of the slice s through wireless links to achieve low wireless communication latency, after that, through the virtual links of s can user u connect to VNF v. We denote the nearest VNF to user u at time slot t as $V_u(t)$, where $V_u(t)$ can be obtained by

$$V_u(t) = \arg\min_{v \in \mathcal{V}_s} \{(X_{h_v} - X_u(t))^2 + (Y_{h_v} - Y_u(t))^2\}. \quad (1)$$

Moreover, we consider that different slices have different latency performances of their virtual links to support different preferences of latency of users, which can be realized by allocating better network channels to support latency-sensitive services based on the physical links [13], [14]. We denote the latency for user $u \in \mathcal{U}_s$ to access the nearest VNF in slice s as P_s, the latency between each of two directly connected VNFs as $\eta_s, \forall s \in \mathcal{S}$, thus, $l^i_{u,v}(t)$ can be calculated by

$$l^i_{u,v}(t) = P_s + \eta_s |\Omega(V_u(t), v)|, \forall u \in \mathcal{U}_s, \forall v \in \mathcal{V}_s, \forall s, \forall i, \forall t, \quad (2)$$

where $|\Omega(V_u(t), v)|$ denotes the number of VNFs on the shortest path between $V_u(t)$ and v in the topology of slice s.

C. Problem Formulation

We aim to improve the QoE of users by predicting the potential anomalies and optimizing the resource provisioning strategies of slices when no anomaly occurs, to this end, we define $\Delta_s(t) \in \{0, 1\}$ to indicate the alarm status of slice s at time slot t, where "1" indicates alarm and "0" indicates not. Thus, $\Delta_s(t)$ can be calculated by

$$\Delta_s(t) = \begin{cases} 0, \sum_{v \in \mathcal{V}_s} R^i_{u,v}(t) \geq R^i_u(t), l^i_{u,v}(t) \leq L^i_u(t); \\ 1, \quad otherwise; \end{cases} \quad (3)$$
$$\forall u \in \mathcal{U}_s, \forall v \in \mathcal{V}_s, \forall s \in \mathcal{S}, \forall i, \forall t.$$

When a slice runs normally and does not occur alarm, the QoE of the users should be related to the allocated resources from the slice and the corresponding latencies of resources provided. Otherwise, if the slice occurs anomaly, the slice has to adjust the VNFs and will affect the QoE of users. To measure the QoE of users, we define the NPS of user u at time slot t as $\text{NPS}_u(t)$. Moreover, for each time slot and resource type i, we denote the user satisfaction of a resource unit as $\alpha^i > 0$ when the user requests are satisfied by slices, and the user cost as $\gamma^i < 0$ when the slices occur alarm. Thus, the NPS of user u at time slot t can be calculated as

$$\text{NPS}_u(t) = \frac{1}{I} \sum_{i=1}^{I} \{(1 - \Delta_s(t))\ \alpha^i \sum_{v \in \mathcal{V}_s} \frac{R^i_{u,v}(t)}{l^i_{u,v}(t)} + \Delta_s(t)\ \ \gamma^i\ R^i_u(t)\}, \forall u \in \mathcal{U}_s, \forall t. \quad (4)$$

We aim to maximize the long-term system NPS which is the sum of all the users in the system, denoting indicator variable as $\Pi(\cdot) \in \{0, 1\}$ where "1" means "$\cdot$" holds and "0" means not, thus, the problem can be formulated as

$$\max_{\Delta_s(t), R^i_{u,v}(t)} \quad \sum_t \sum_{u \in \mathcal{U}} \text{NPS}_u(t), \quad (5a)$$

$$\sum_{u \in \mathcal{U}} R^i_{u,v}(t) \leq R^i_v, \forall v \in \mathcal{V}_s, \forall i, \forall t, \quad (5b)$$

$$\sum_{s \in \mathcal{S}} \sum_{v \in \mathcal{V}_s} \Pi(h_v = n) R^i_v \leq R^i_n, \forall n \in \mathcal{N}, \forall i, \quad (5c)$$

$$\mathcal{U}_s \cap \mathcal{U}_{s'} = \emptyset, \forall s, s' \in \mathcal{S}, \quad (5d)$$

$$\Delta_s(t) \in \{0, 1\}, \forall s \in \mathcal{S}, \forall t. \quad (5e)$$

Here, constrain (5b) ensures that the allocated resources for all serving users of VNF v cannot be more than the total resources of the VNF; constrain (5c) indicates the total resources of hosted VNFs of ES n cannot exceed the resource of the ES; constrain (5d) ensures one user can only request one slice for resource provisioning.

Remark. From the formulated problem we can see that when the slices run normally, the NPSs of users depend on the allocated resources and latencies, while the former is satisfied and equal to the requested resources; otherwise, when the slices occur anomalies, the NPSs of users equal to the cost of their unsatisfied resources and the slices have to adjust the VNFs until they can satisfy the users' requests. According to our previous research [15], adjusting network slices based on the prediction of future information in advance can introduce lower user latencies. Thus, based on these considerations, we propose to predict the anomalies of the slices based on the user behaviors, which will be utilized to improve the long-term system NPS in two directions as: *1)* When the slices are predicted to occur anomalies, the slices can adjust the VNFs in advance to achieve much lower latencies and thus higher normally running NPS afterward; *2)* Otherwise, the slices will adjust the resource allocation strategies to reduce the resource provisioning latency.

III. Slice Anomaly Prediction and Resource Allocation Framework

In this part, we first introduce the process for learning users' behavior to get the future prediction of users' requests and positions. After that, we propose an anomaly prediction and resource allocation algorithm.

A. User Behavior Prediction

To learn the users' behaviors, we first collect the users' historical data including the requested resources, latencies, and positions of users for several time slots. Without loss of generality, we assume that all the users' behavior data in the considered system are independent and identically distributed. After the collection process, we use the long short-term memory (LSTM) model for user behavior learning and then further predict users' requests and positions. Particularly, in addition

Algorithm 1 User Behaviour Prediction Algorithm

1: **Input**: φ, σ, ϕ, $\mathcal{U}$.
2: **Initialize**: The historical dataset of user u as $D_u^{his} = \emptyset$.
3: **for** $u \in \mathcal{U}$ **do**
4: **for** $t = \{1, \ldots, \varphi\}$ **do**
5: $D_u^{his} \leftarrow D_u^{his} + \{X_u(t), Y_u(t), \cup_{i=1}^{I}\{R_u^i(t), L_u^i(t)\}$.
6: **end for**
7: Get the training samples $\mathrm{X}_{u,1}, \ldots, \mathrm{X}_{u,\varphi-\sigma}$ and testing samples $\mathrm{Y}_{u,1}, \ldots, \mathrm{Y}_{u,\varphi-\sigma}$ according to (6) and (7), respectively.
8: **for** Each train epoch **do**
9: With $\mathrm{X}_{u,1}, \ldots, \mathrm{X}_{u,\varphi-\sigma}$, fit the LSTM model, get the output as $\hat{\mathrm{Y}}_{u,1}, \ldots, \hat{\mathrm{Y}}_{u,\varphi-\sigma}$.
10: Calculate the MSE loss between $\hat{\mathrm{Y}}_{u,1}, \ldots, \hat{\mathrm{Y}}_{u,\varphi-\sigma}$ and $\mathrm{Y}_{u,1}, \ldots, \mathrm{Y}_{u,\varphi-\sigma}$.
11: Update the parameters of LSTM by Adam optimizer and backward propagation method.
12: **end for**
13: **end for**
14: **for** $u \in \mathcal{U}$ **do**
15: **for** $t > \varphi$ **do**
16: Get the historical stack sequence with length σ as $H_u(t) = \{X_{u,t-\sigma+1}, X_{u,t-\sigma+2}, \ldots, X_{u,t}\}$.
17: Input the historical stack $H_u(t)$ to the well-trained LSTM model to obtain the prediction $P_u(t+1)$.
18: **end for**
19: **end for**
20: **Output**: $P_u(t+1), \forall t > \varphi, \forall u \in \mathcal{U}$.

to the external recurrent neural network loops, the LSTM model introduces the concept of gates and adds internal cell loops to solve the long-distance dependency issue, thus, it can significantly reduce learning difficulty by providing long-term memory for valuable information. We elaborate the learning and prediction process as Algorithm 1. Specifically, for each user $u \in \mathcal{U}$ we consider collecting φ time slots' historical behavior data for model training, which can be expressed as $D_u^{his} = \{X_u(1), Y_u(1), \cup_{i=1}^{I}\{R_u^i(1), L_u^i(1)\}, \ldots, X_u(\varphi), Y_u(\varphi), \cup_{i=1}^{I}\{R_u^i(\varphi), L_u^i(\varphi)\}\}$ (Lines 4-6). Afterward, the historical dataset of user u can be split into training and testing data samples (Line 7). Denote each training sample consists of σ time slots' user data, based on D_u^{his}, the $\varphi - \sigma$ training samples can be expressed by

$$\begin{aligned}\mathrm{X}_{u,1} = \{&X_u(1), Y_u(1), \cup_{i=1}^{I}\{R_u^i(1), L_u^i(1)\}, \ldots, \\ &X_u(\sigma), Y_u(\sigma), \cup_{i=1}^{I}\{R_u^i(\sigma), L_u^i(\sigma)\}\}, \\ &\ldots, \\ \mathrm{X}_{u,\varphi-\sigma} = \{&X_u(\varphi-\sigma), Y_u(\varphi-\sigma), \cup_{i=1}^{I}\{R_u^i(\varphi-\sigma), \\ &L_u^i(\varphi-\sigma)\}, \ldots, X_u(\varphi-1), Y_u(\varphi-1), \\ &\cup_{i=1}^{I}\{R_u^i(\varphi-1), L_u^i(\varphi-1)\}\},\end{aligned} \tag{6}$$

and the corresponding testing samples can be expressed as

$$\begin{aligned}\mathrm{Y}_{u,1} = \{&X_u(\sigma+1), Y_u(\sigma+1), \cup_{i=1}^{I}\{R_u^i(\sigma+1), \\ &L_u^i(\sigma+1)\}\}, \\ &\ldots, \\ \mathrm{Y}_{u,\varphi-\sigma} = \{&X_u(\varphi), Y_u(\varphi), \cup_{i=1}^{I}\{R_u^i(\varphi), L_u^i(\varphi)\}\}.\end{aligned} \tag{7}$$

The training samples and testing samples then will be sent to the LSTM model for updating the parameters by the backward propagation algorithm based on the MSE loss between the testing data and model prediction (Lines 8-11). When the model is well-trained, in the following time slots $t > \varphi$, we utilize the model for future user position and requests prediction (Lines 16-17). As a result, the output of the algorithm $P_u(t+1), \forall t > \varphi$ includes the future prediction of the user's position information and requested resources/ latencies, expressed as

$$\begin{aligned}P_u(t+1) = \{&\hat{X}_u(t+1), \hat{Y}_u(t+1), \cup_{i=1}^{I}\{\hat{R}_u^i(t+1), \\ &\hat{L}_u^i(t+1)\}\}.\end{aligned} \tag{8}$$

Thus, based on the prediction of users' information $P_u(t+1)$, we can locate the potential anomalies and decide the resource allocation strategies of slices.

Algorithm 2 Anomaly Prediction and Resource Allocation Algorithm

1: **Input**: I, $\mathcal{U}$, $P_u(t+1), \forall t > \varphi$, $\mathcal{S}$.
2: **Initialize**: The future anomaly flag of slice s as $\hat{\Delta}_s(t+1) = 0$, $R_{u,v}^i(t+1) = 0$, the remaining resource of VNF v as $R_v^{i,rem}(t+1) = R_v^i$, the unsatisfied resource $R_u^{i,ust}(t+1) = \hat{R}_u^i(t+1)$. .
3: **for** $s \in \mathcal{S}$ **do**
4: **for** $i = 1, \ldots, I$ **do**
5: Sort the user list according to users' requested amount of resource type i $\hat{R}_u^i(t+1)$ with a descending order, get the sorted user list $\mathcal{U}_s'$, where the first user has the highest $\hat{R}_u^i(t+1)$.
6: **for** $u \in \mathcal{U}_s'$ **do**
7: Calculate latencies from the user to all the VNFs $\mathcal{V}_s$ according to (2), denoted as $L_{u,s}^i(t+1) = \underset{v \in \mathcal{V}_s}{\cup} \hat{l}_{u,v}^i(t+1)$.
8: **if** $\min\{L_{u,s}^i(t+1)\} > \hat{L}_u^i(t+1)$ **then**
9: $\hat{\Delta}_s(t+1) = 1$.
10: **else**
11: Filter the VNFs to get those with $\hat{l}_{u,v}^i(t+1) \leq \hat{L}_u^i(t+1)$ and sort according to the latency with a descending order, denote the new VNFs as $\mathcal{V}_u^i(t+1)$.
12: **for** $v \in \mathcal{V}_u^i(t+1)$ **do**
13: $R_{u,v}^i(t+1) \leftarrow \min\{R_u^{i,ust}(t+1), R_v^{i,rem}(t+1)\}$.
14: $R_v^{i,rem}(t+1) \leftarrow R_v^{i,rem}(t+1) - R_{u,v}^i(t+1)$
15: $R_u^{i,ust}(t+1) \leftarrow R_u^{i,ust}(t+1) - R_{u,v}^i(t+1)$.
16: **if** $R_u^{i,ust}(t+1) = 0$ **then**
17: Break.
18: **end if**
19: **end for**
20: **if** $R_u^{i,ust}(t+1)\ != 0$ **then**
21: $\hat{\Delta}_s(t+1) = 1$.
22: **end if**
23: **end if**
24: **end for**
25: **end for**
26: **end for**
27: **Output**: $\hat{\Delta}_s(t+1), R_{u,v}^i(t+1)$.

B. Slice Anomaly Prediction and Resource Allocation

After obtaining the prediction information $P_u(t+1)$, at time slot t we can predict if there will be potential slice anomalies at time slot $t+1$. If the resource provisioning cannot satisfy the users' requests, the slices trigger an anomaly alarm at time slot t and the slices tend to adjust the VNFs in advance; otherwise,

the slices optimize the resource allocation strategies of VNFs to improve the system NPS, the detailed process of slice anomaly prediction and resource allocation is described as Algorithm 2.

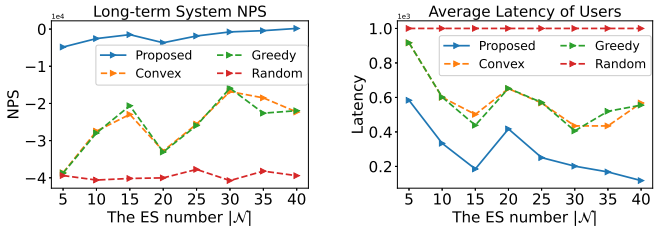
Specifically, at each time slot $t > \varphi$ we calculate if users' future requests can be satisfied or not with the prediction information of time slot $t+1$. To this end, we initialize the remaining resource of VNF v as the total amount of resources $R_v^i, \forall v \in \mathcal{V}_s, \forall i$, the unsatisfied resources of user u as the predicted requested resources $\hat{R}_u^i(t+1), \forall u$ (Line 2). After that, we first order the users by their predicted requested amount of resource type i $\hat{R}_u^i(t+1), \forall u \in \mathcal{U}_s$ with a descending order and try dispatching the resources for users in turn (Line 5). Afterwards, for each user u in the sorted user list $\mathcal{U}_s'$, we calculate the future latencies from user u to the all the VNFs $v \in \mathcal{V}_s$ based on the predicted position information to get $\bigcup_{v \in \mathcal{V}_s} \hat{l}_{u,v}^i(t+1)$ (Line 7). Once all the VNFs cannot support the preferred latencies from the user u, i.e., $\hat{l}_{u,v}^i(t+1) > \hat{L}_u^i(t+1), \forall v \in \mathcal{V}_s$, the slice s predicts an alarm as there exist at least one user's requested resources cannot be satisfied (Lines 8-10). Otherwise, we filter the VNFs that satisfy the preferred latency and search from the highest latency in turn to calculate the possible allocation of resources based on the remaining resources of the VNFs and the requested resources from users, until the user's requested resources get fulfilled (Lines 11-15). If all the VNFs cannot meet the requested resources, the slice should occur an alarm for the potential anomaly which indicates that under the most extreme case, at least one user's requests cannot be satisfied by the slice, and the slice s should be an abnormal slice at time slot $t+1$.

IV. Simulation Results

In this part, we introduce the simulation setting and present the evaluation results to demonstrate the performance of our proposed framework.

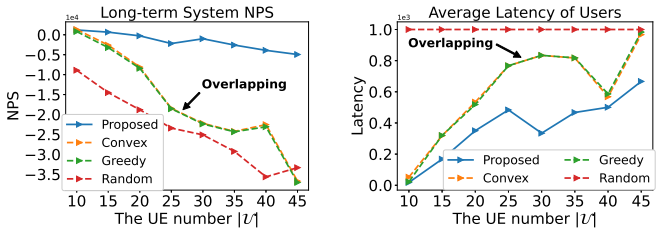
A. Setup

To evaluate the proposed framework, we consider there exists 50 UEs dynamically move across a square with 100×100 M^2 and 10 BSs are geographically evenly distributed. The UEs request three types of resources, i.e., CPU, RAM, and disk, and we have $I = 3$. The total resources of the 10 ESs are uniformly set as 8 CPU cores, 32 Gbits RAM, and 256 GBits disk. To mimic the dynamically changing user requests and positions, at the beginning of the simulation, we consider the users randomly requested resource intervals of CPU, RAM, and disk as $[0.01, 2]$, $[0.01, 4]$, and $[0.01, 8]$, respectively; the requested latencies of UEs randomly ranges from 10 ms to 50 ms, and the positions of UEs are set as $X_u(t) \in [0, 100], Y_u(t) \in [0, 100]$. Based on the requests from the users, the network slices are initialized according to different types of resources and consist of several VNFs hosted by different ESs. We use the **networkx** module of Python to generate the topologies of ESs and the slices. Considering the slices have different latency preferences, we set the radio access latency from the UEs to the nearest VNF of the slices randomly from 1 ms to 3 ms, and the



(a) The long-term system NPS. (b) The average latency of users.

Fig. 2. The long-term system NPS and the average latency of users of the considered schemes versus $|\mathcal{N}|$.



(a) The long-term system NPS. (b) The average latency of users.

Fig. 3. The long-term system NPS and the average latency of users of the considered schemes versus $|\mathcal{U}|$.

latency between each two connected VNFs from 3 to 5 ms [13], [16]. After the initialization, the users periodically change their requested resources, latencies, as well as positions based on their previous time slot's behavior. The user satisfaction value α^i and user cost γ^i is set as 10 and -20, respectively.

B. Baseline and Performance Metrics

To evaluate the performances of the proposed scheme, we compare our proposed scheme with baselines as: *1)* **Greedy** resource provisioning scheme, in which the slices search for the VNFs with the most available resources in turn and try to satisfy the requested resources with preferred latencies; *2)* **Convex** resource provisioning scheme, where the slices allocate the requested resources with a convex optimization method [17] for users from all VNFs that can provide preferred latencies by linear programming; *3)* **Random** resource provisioning scheme, where the slices allocate the requested resources randomly for users from all VNFs with preferred latencies. Different from these schemes, we not only use the prediction of users' requests to decide the resource provisioning strategy in advance but also simultaneously consider satisfying the requested resource and reducing the service latency to improve users' QoE. As a result, the performance indicators in this paper include the long-term system NPS and the average latency of users over time slots.

C. Evaluation Results

We first compare the long-term system NPS of the considered schemes versus different numbers of ESs, as shown in Fig. 2. Specifically, from Fig. 2(a) we can observe the system NPS of the considered schemes gradually increases with the numbers of ESs, while the proposed scheme and the random scheme have relatively stable tendencies. The system NPS of all considered schemes are lower than 0 since most slices have abnormal users

whose requested resources/latencies cannot be satisfied. When the number of ESs equals 20 and 25, we find the requests from UEs have high latency requirements and the slices have fewer candidate VNFs that can satisfy the requested latencies than other cases, leading to worse resource provisioning, much more abnormal slices, and finally lower system NPS. The proposed scheme with users' behavior prediction can achieve higher system NPS compared to other baselines thanks to the lower user latencies and more reasonable resource provisioning strategy. From Fig. 2(b) we can see the system latency of the proposed scheme, convex scheme, and greedy scheme gradually decrease with the numbers of ESs in most cases, while the random scheme keeps stable, which is because when we have more ESs, the UEs can have more candidate VNFs for satisfying their requests and thus can achieve lower latencies. Overall, the proposed scheme can significantly improve the long-term system NPS and reduce the average latency of users compared to other baseline schemes for all ESs.

Further, with different numbers of UEs, we compare the long-term system NPS and the average latency of users of the considered schemes as shown in Fig. 3. From Fig. 3(a) we can see the system NPS decreases with the number of UEs, while the proposed scheme has a stable tendency. The convex and greedy schemes have very close behaviors to each other and the two curves nearly overlap. When we have more UEs, the competition among UEs tends to exacerbate since the available resources of VNFs are limited and more UEs cannot get the requested resources, resulting in the increase of abnormal slices and finally lower long-term system NPS. From Fig. 3(b) we can see the average latency of users increases with the number of UEs, while the proposed scheme has the lowest average latency, followed by the greedy scheme, the convex scheme, and the random scheme. As a result, the proposed scheme outperforms the considered schemes in improving the long-term system NPS and reducing the average latency of users for all UEs.

V. Conclusion

In this paper, we have investigated the resource provisioning of network slices and focused on improving the QoE of users by detecting the potential anomalies of slices based on the historical behaviors of users. We have modeled the whole process for slice resource provisioning and formulated the problem as maximizing the long-term system NPS. To solve the problem, we have designed a framework that learns the users' behavior for slice anomaly prediction and resource allocation. Finally, simulation results have demonstrated that our proposed scheme outperforms the considered baselines in improving the long-term NPS and reducing the average latency of users.

Acknowledgments

This research work is partially supported by the European Union's Horizon 2020 Research and Innovation Program through the aerOS project under Grant No. 101069732; the Business Finland 6Bridge 6Core project under Grant No. 8410/31/2022; the European Union's HE research and innovation program HORIZON-JUSNS-2022 under the 6GSandbox project (Grant No. 101096328); and the Research Council of Finland 6G Flagship Programme under Grant No. 346208. This research was also partially conducted at ICTFICIAL Oy. The paper reflects only the authors' views, and the European Commission bears no responsibility for any utilization of the information contained herein.

References

[1] O. Chukhno, N. Chukhno, G. Araniti, C. Campolo, A. Iera, and A. Molinaro, "Placement of social digital twins at the edge for beyond 5g iot networks," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 927–23 940, July 2022.

[2] F. Rezazadeh, H. Chergui, L. Blanco, L. Alonso, and C. Verikoukis, "A collaborative statistical actor-critic learning approach for 6g network slicing control," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2021, pp. 1–6.

[3] T. K. Rodrigues and N. Kato, "Network slicing with centralized and distributed reinforcement learning for combined satellite/ground networks in a 6g environment," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 104–110, Apr. 2022.

[4] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. M. Leung, "Cooperative computation offloading for multi-access edge computing in 6g mobile networks via soft actor critic," *IEEE Transactions on Network Science and Engineering*, Apr. 2021.

[5] R. A. Addad, D. L. C. Dutra, T. Taleb, and H. Flinck, "Toward using reinforcement learning for trigger selection in network slice mobility," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2241–2253, July 2021.

[6] R. A. Addad, T. Taleb, H. Flinck, M. Bagaa, and D. Dutra, "Network slice mobility in next generation mobile systems: Challenges and potential solutions," *IEEE Network*, vol. 34, no. 1, pp. 84–93, Sept. 2020.

[7] Z. Ming, X. Li, C. Sun, Q. Fan, X. Wang, and V. C. M. Leung, "Dependency-aware hybrid task offloading in mobile edge computing networks," in *Proc. IEEE International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, Dec. 2021, pp. 225–232.

[8] X. Wang, C. Wang, X. Li, V. C. M. Leung, and T. Taleb, "Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9441–9455, Apr. 2020.

[9] W. Wang, Q. Chen, X. He, and L. Tang, "Cooperative anomaly detection with transfer learning-based hidden markov model in virtualized network slicing," *IEEE Communications Letters*, vol. 23, no. 9, pp. 1534–1537, July 2019.

[10] W. Wang, C. Liang, Q. Chen, L. Tang, H. Yanikomeroglu, and T. Liu, "Distributed online anomaly detection for virtualized network slicing environment," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 11, pp. 12 235–12 249, Dec. 2022.

[11] F. Zhou, P. Yu, L. Feng, X. Qiu, Z. Wang, L. Meng, M. Kadoch, L. Gong, and X. Yao, "Automatic network slicing for iot in smart city," *IEEE Wireless Communications*, vol. 27, no. 6, pp. 108–115, Jan. 2020.

[12] X. Vasilakos, N. Nikaein, D. H. Lorenz, B. Koksal, and N. Ferdosian, "Integrated methodology to cognitive network slice management in virtualized 5g networks," *arXiv preprint arXiv:2005.04830*, 2020.

[13] L. Zanzi, V. Sciancalepore, A. Garcia-Saavedra, H. D. Schotten, and X. Costa-Pérez, "Laco: A latency-driven network slicing orchestration in beyond-5g networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 667–682, Oct. 2020.

[14] S. O. Oladejo and O. E. Falowo, "Latency-aware dynamic resource allocation scheme for multi-tier 5g network: A network slicing-multitenancy scenario," *IEEE Access*, vol. 8, pp. 74 834–74 852, Apr. 2020.

[15] H. Yu, Z. Ming, C. Wang, and T. Taleb, "Network slice mobility for 6g networks by exploiting user and network prediction," in *Proc. IEEE International Conference on Communications (ICC)*, May 2023.

[16] K. Boutiba, M. Bagaa, and A. Ksentini, "Radio resource management in multi-numerology 5g new radio featuring network slicing," in *Proc. IEEE International Conference on Communications (ICC)*, May 2022, pp. 359–364.

[17] X. Zhang, H. Zhang, W. Du, K. Long, and A. Nallanathan, "Irs empowered uav wireless communication with resource allocation, reflecting design and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 21, no. 10, pp. 7867–7880, Oct. 2022.